StreamMOS: Streaming Moving Object Segmentation with Multi-View Perception and Dual-Span Memory

Zhiheng Li, Yubo Cui, Jiexi Zhong, and Zheng Fang*

***Abstract*—Moving object segmentation based on LiDAR is a crucial and challenging task for autonomous driving and mobile robotics. Most approaches explore spatio-temporal information from LiDAR sequences to predict moving objects in the current frame. However, they often focus on transferring temporal cues in a single inference and regard every prediction as independent of others. This may lead to inconsistent segmentation results for the same object across different frames. To solve this issue, we propose a streaming network with a memory mechanism, called StreamMOS, to build the association of features and predictions among multiple inferences. Specifically, we utilize a short-term memory to convey historical features, which can be regarded as spatial priors of moving objects and are used to enhance current inference by temporal fusion. Meanwhile, we build a long-term memory to store previous predictions and exploit them to refine current forecasts at the voxel and instance levels through voting. Besides, we apply multi-view encoder with cascaded projection and asymmetric convolution to extract motion feature of objects in different representations. Extensive experiments validate that our algorithm gets competitive performance on SemanticKITTI and Sipailou Campus datasets.**

***Index Terms*—Semantic Scene Understanding; Deep Learning Methods; Computer Vision for Transportation**

I. Introduction

On urban roads, there are often many dynamic objects with variable trajectories, such as vehicles and pedestrians, which create the collision risk for autonomous vehicles. Meanwhile, these moving objects will cause errors in simultaneous localization and mapping (SLAM) [1], as well as pose challenges for obstacle avoidance [2] and path planning [3]. As a result, online moving object segmentation (MOS) based on LiDAR points has become a crucial task in multiple fields. However, owing to the unordered and sparse nature of LiDAR points, MOS still faces some challenges, especially difficulty in perceiving moving objects with sparse points at a distance.

Manuscript received: July, 25, 2024; Revised November, 13, 2024; Accepted December, 5, 2024.

This paper was recommended for publication by Editor Cesar Cadena Lerma upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants 62073066, in part by the Fundamental Research Funds for the Central Universities under Grant N2226001, and in part by 111 Project under Grant B16009. (*Corresponding author: Zheng Fang*)

The authors are all with Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China. Zhiheng Li and Zheng Fang are also with the National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China and also with Key Laboratory of Data Analytics and Optimization for Smart Industry, Ministry of Education, Northeastern University, Shenyang 110819, China. (e-mail: fangzheng@mail.neu.edu.cn)

The code will be open at https://github.com/NEU-REAL/StreamMOS.git.

Digital Object Identifier (DOI): see top of this page.

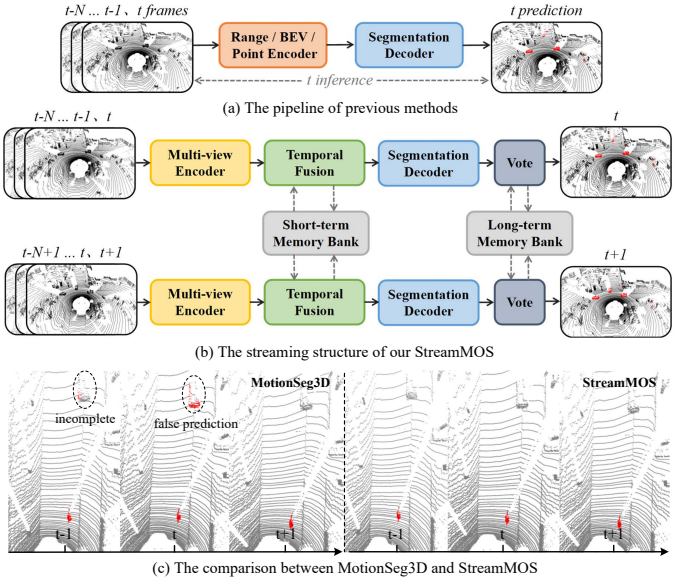


Fig. 1. **Pipeline comparison of moving object segmentation approaches.** We compare the structure of proposed StreamMOS with previous methods in (a) and (b). Meanwhile, the segmentation results obtained by our method achieve better spatial integrity and temporal continuity in (c).

To tackle the above problem, the mainstream strategy is to exploit spatio-temporal information from LiDAR sequences. For instance, Chen et al. [4] generate residual image in range view (RV), which reflects spatial positions of moving objects in each frame and can be utilized to perform temporal fusion to infer motion states of objects. Based on the RV projection in [4], Sun et al. [5] adopt motion-guided attention to better explore temporal motion cues from residual images. Besides, some works [6], [7] attempt to map point clouds to bird's eye view (BEV) and ensure consistent object size and movement. Recently, Wang et al. [8] process LiDAR sequences directly via 4D convolution to construct temporal associations while adding instance detection to promote segmentation integrity.

However, as displayed in Fig. 1(a), these methods focus on temporal fusion in a single inference and make independent predictions for each frame, leading to inconsistent results for the same object at different moments (in Fig. 1(c)). Despite Mersch et al. [9] leveraging a binary Bayes filter to combine multiple predictions, it still ignores information transmission at feature level, which supplies rich spatial context to the next inference. Thus, we present a "streaming" structure as shown in Fig. 1(b), which regards historical feature as a strong prior and exploits it to guide the current inference. Meanwhile, the past predictions are stored in long-term memory and utilized

to suppress false predictions. In this way, we construct robust correlations in multiple inferences and fully explore temporal information to ensure consistent results in different frames.

To implement the idea of streaming, we propose a moving object segmenter, named *StreamMOS*, which captures object motion across multiple views and adopts dual-span memory to transfer historical information. Specifically, different from previous works that map point clouds to one view, we argue that various viewpoints provide more holistic observations of dynamic objects. Thus, we propose a multi-view encoder that uses a cascaded structure to iteratively get dense appearance from RV and perceive intuitive motion on BEV, resulting in more discriminative features of moving objects (in Fig. 3(b)). Meanwhile, during BEV encoding, we introduce asymmetric convolution with decoupled strategy to better capture vertical and horizontal motion. Then, we use attention mechanism to implement temporal fusion that aligns features from different times and conveys spatial prior to current inference. Besides, due to the inherent uncertainty of neural networks, the output of segmentation decoder could be inconsistent across frames (Fig. 1(c)). To solve this issue, we propose voting mechanism as post-processing to optimize predicted labels. Its core idea is to statistically analyze long-term motion states at voxel and instance levels, and then select the most likely state to update raw point-wise forecasts. In this way, the previous results can be used to refine current predictions, enhancing the temporal continuity and spatial completeness of segmentation together.

In summary, the contributions of our work are as follows:

- We present a novel streaming framework called StreamMOS, which exploits short-term and long-term memory to construct associations among inferences and improve the integrity and continuity of predictions in MOS task.
- We propose a multiple projection architecture to capture object motion and complete appearance from different views. We introduce a multi-level voting mechanism to refine segmentation results for every voxel and instance.
- The experimental results confirm that our StreamMOS outperforms the previous state-of-the-art on the test sets of SemanticKITTI by 1.1% IoU and Sipailou Campus by 1.7% IoU, while achieving competitive running time.

II. Related Work

A. Geometric-based Algorithms

The initial LiDAR-based MOS algorithms can be referred to as the *geometric-based approaches*, which typically build the map in advance and remove any dynamic objects through estimating occupancy probability and determining visibility. For example, Schauer et al. [10] proposed a ray casting-based approach that counted the hits and misses of scans to update the occupancy situation of the grid map. Afterwards, Pagad et al. [11] constructed an occupancy octree map and proposed a probability update mechanism to obtain clean point clouds by considering the occupancy history. Despite getting promising results, [10], [11] suffer extensive computational burden due to the ray casting and updating voxel one by one. To improve efficiency, several visibility-based [12], [13], [14] algorithms have been developed. Pomerleau et al. [12] identified moving objects by checking whether the points of the pre-built map are occluded by the points in the query frame. Meanwhile, to avoid mismarked ground points as dynamic reported in [12], Kim et al. [13] retained ground points from removed points using a multi-resolution reverting algorithm. Moreover, Lim et al. [15] introduced a visibility-free approach that removed moving traces by computing pseudo occupancy ratio between the query scan and the submap in each grid. Building on [15], Zhang et al. [16] proposed a height coding descriptor, while Lim et al. [17] introduced instance segmentation to maximize the preservation of static points. Although the above methods clean maps well, they are performed offline due to requiring a prior map, making them unsuitable for real-time applications.

B. Learning-based Algorithms

Recently, many studies have focused on utilizing *learning-based approaches* to eliminate dynamic objects online, which take only consecutive frame point clouds as input rather than a pre-built map. Meanwhile, according to data representation, these algorithms can be categorized into projection-based and point-based methods. The former converts point clouds into bird's eye view (BEV) or range view (RV) images, while the latter processes 3D raw points directly.

Specifically, for *point-based algorithms*, Mersch et al. [9] adopted sparse 4D convolutions to process a series of LiDAR scans and predicted moving objects in each frame. They also employed a binary Bayes filter to fuse multiple predictions in a sliding window. Subsequently, Kreutz et al. [18] proposed an unsupervised approach to address MOS task in stationary LiDAR and viewed it as a multivariate time series clustering problem. Lately, Wang et al. [8] introduced InsMOS to unify detection and segmentation of moving objects into a network, so that the instance cues can be used to improve segmentation integrity. Although they achieved promising performance, the feature extraction of numerous points in [8] may lead to high computational costs.

Compared to the mentioned approaches, *projection-based algorithms* [4], [5], [19], [6], [7] are generally more efficient owing to handling ordered and dense data. For instance, Chen et al. [4] mapped LiDAR scans into spherical coordinates and generated residual images to extract dynamic information in sequence. Sun et al. [5] used a dual-branch network to encode spatio-temporal information and mitigated boundary blurring problem with a point refinement module. In addition, Kim et al. [19] achieved higher performance by using extra semantic features. In contrast to the RV projection, Mohapatra et al. [6] and Zhou et al. [7] utilized BEV projection to obtain a more intuitive motion representation, but the serious loss of spatial information still limited performance. To solve this issue, our StreamMOS captures object motion from multiple views in a series manner, allowing for complete observation of objects. We also build memory banks to convey historical knowledge, resulting in consistent segmentation across a long sequence.

III. Methodology

A. Framework Overview

LiDAR-based MOS aims to determine the motion state of each point in the current scan based on the multi-frame point

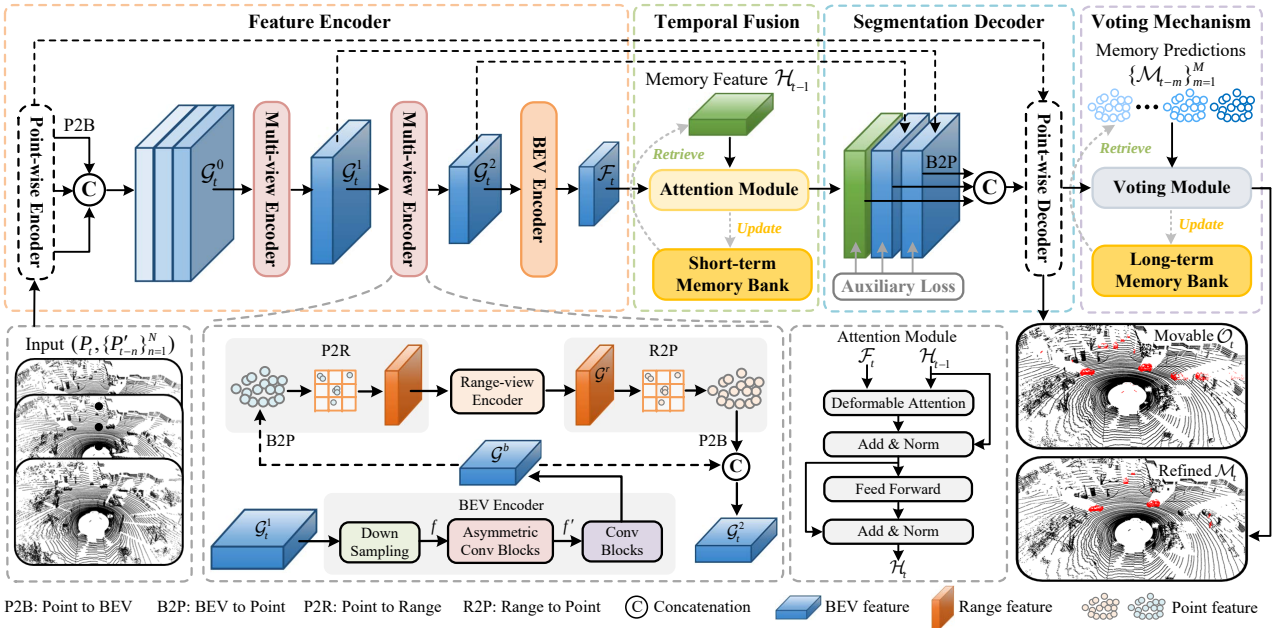


Fig. 2. **The overall architecture of StreamMOS.** (a) Feature encoder adopts a point-wise encoder to extract point features and project them into BEV. Then, the multi-view encoder with cascaded structure and asymmetric convolution is applied to encode motion features from different views. (b) Temporal fusion utilizes an attention module to propagate memory feature to the current inference. (c) Segmentation decoder with parameter-free upsampling exploits multi-scale features to predict class labels. (d) Voting mechanism leverages memory predictions to optimize the motion state of each 3D voxel and instance.

clouds $\{\mathcal{P}_{t-n}\}_{n=0}^{N}$. To this end, existing methods first adopt the relative pose transformations $\{\mathcal{T}_{t-n\to t}\}_{n=1}^{N}$ provided by the LiDAR odometry to project historical scans $\{\mathcal{P}_{t-n}\}_{n=1}^{N}$ into ego car coordinate system of the current scan $\mathcal{P}_t$ and get $\{\mathcal{P}'_{t-n}\}_{n=1}^{N}$. Then, they usually feed $\mathcal{P}_t$ and $\{\mathcal{P}'_{t-n}\}_{n=1}^{N}$ into a network Ψ to fuse spatio-temporal information and predict classification results $\mathcal{M}_t \in \mathbb{R}^{V\times 3}$ for all points in $\mathcal{P}_t$, where $V \times 3$ refers to the probability that V points belong to three categories, including unknown, static and moving states.

Different from previous approaches that focus on temporal fusion in a single inference, we extra consider the association among multiple inferences and apply historical feature $\mathcal{H}_{t-1}$ and predictions $\{\mathcal{M}_{t-m}\}_{m=1}^{M}$ to raise the quality of current inference. Thus, our method formulates MOS task as follows:

$$\mathcal{M}_t = \Psi(\mathcal{P}_t, \{\mathcal{P}'_{t-n}\}_{n=1}^{N}, \mathcal{H}_{t-1}, \{\mathcal{M}_{t-m}\}_{m=1}^{M}) \quad (1)$$

where N, M are the number of historical LiDAR frames and forecasts. Meanwhile, the details of our network are shown in Fig. 2. Specifically, given a series of scans, our StreamMOS first leverages the multi-view encoder to capture motion cues from the viewpoints of BEV and RV. Thereafter, we can get a motion feature $\mathcal{F}_t$ that reflects spatial information of moving objects in the current frame. Then, we use a temporal fusion module to combine $\mathcal{F}_t$ with historical feature $\mathcal{H}_{t-1}$ retained in short-term memory. By doing this, some prior information can be transferred to the current inference and further utilized to decode movable objects $\mathcal{O}_t$ and coarse motion state $\mathcal{C}_t$ for all points. Finally, we employ a voting mechanism to update $\mathcal{C}_t$ using historical results $\{\mathcal{M}_{t-m}\}_{m=1}^{M}$ stored in long-term memory and instance information derived from $\mathcal{O}_t$, thereby yielding the refined prediction $\mathcal{M}_t$.

B. Multi-projection Feature Encoder

1) Preliminaries: Unlike the existing methods that project point clouds into a single view, such as BEV [7] or RV [19], we believe that mapping points to these views simultaneously could capture more complete appearance and obvious motion cues of dynamic objects. Meanwhile, as shown at the bottom of Fig. 2, the points could be considered as the intermediate carrier to transfer information between different perspectives. For this purpose, we adopt *Point-to-BEV* (P2B) and *Point-to-Range* (P2R) to map point features into 2D plane, while using *BEV-to-Point* (B2P) and *Range-to-Point* (R2P) to collect the point features from multiple planes. To be specific, assuming that the k^{th} 3D point in $\mathcal{P}_t$ is denoted as $p_k^{3D} = (x_k, y_k, z_k)$, the P2B projects it into a rectangular 2D grid and obtains its coordinate (u_k^b, v_k^b) in BEV. For the P2R, the point p_k^{3D} with 3D cartesian coordinate is converted into spherical coordinate $p_k^{sph} = (r_k, \theta_k, \phi_k)$ and assigned to the 2D grid in RV with coordinate (u_k^r, v_k^r) [20], where r_k, θ_k, ϕ_k represent distance, zenith and azimuth angle of point p_k^{3D}. The points falling into the same grid undergo max-pooling to aggregate features. For R2P and B2P, the grid features of RV and BEV are allocated to 3D points using bilinear interpolation within nearby grids.

2) Network Structure: In the feature encoder, we first use a lightweight PointNet [21] as point-wise encoder to process point clouds $(\mathcal{P}_t, \{\mathcal{P}'_{t-n}\}_{n=1}^{N})$ and obtain $\mathcal{E}_n \in \mathbb{R}^{V\times C}$ $(n \in \{t-N, ..., t\})$, where C means the number of channels. Then, for the feature of each frame, we adopt P2B to project them into BEV and concatenate them along the channel dimension to get BEV feature $\mathcal{G}_t^0 \in \mathbb{R}^{W^b\times H^b\times (N+1)C}$, where W^b, H^b are the predefined width and height of BEV. Afterwards, we feed $\mathcal{G}_t^0$ into multi-view encoder (MVE) to extract temporal information and capture object motion from different views.

In the lower part of Fig. 2, after downsampling BEV feature $\mathcal{G}_t^l (l \in \{0, 1\})$, we introduce an asymmetric convolution block (ACB) to perceive the movement of objects. As shown in Fig. 3(a), compared to the typical symmetric convolutional kernel (e.g., 3×3), the kernel size of ACB has one side longer

(e.g., 3×5 and 5×3). Besides, it decouples feature extraction into the horizontal and vertical directions, defined as follows:

$$f' = \text{Conv}_{3\times3}(\text{Conv}_h(f) \odot \text{Conv}_v(f)) + f \tag{2}$$

where f and $\odot$ are feature map and concatenation operation. Conv_h and Conv_v mean asymmetric convolutions, which can expand the receptive field and improve perception ability for moving objects since they usually have distinct motion in a specific direction. After that, as displayed in Fig. 2, we apply B2P and P2R to project BEV feature into the range view and then use convolution layer as range-view encoder to generate another motion feature $\mathcal{G}^r$, which is remapped into BEV and combined with $\mathcal{G}^b$ along channel dimension. The multi-view feature interaction is then executed in the next encoder layer.

As a result, complete motion information can be extracted through cascaded projection and encoding within two MVEs and an additional BEV encoder, thereby obtaining a discriminative motion feature $\mathcal{F}_t$. Specially, we illustrate multi-view features of different MVE layers in Fig. 3(b). It proves that MVE can extract consistent object information across various perspectives, while the deeper layer is capable of suppressing noise and preserving clearer motion features.

C. Short-term Temporal Fusion

The purpose of this part is to transfer the memory feature $\mathcal{H}_{t-1}$ from the last inference to the present, so that historical spatial states of objects can be retrieved to guide the network in inferring object motion at time t. To achieve this, we first build short-term memory bank as a bridge to store $\mathcal{H}_{t-1}$ and connect adjacent inferences. Then, since $\mathcal{F}_t$ and $\mathcal{H}_{t-1}$ are not in the same coordinate system, we adopt an attention module with learnable offsets [22] to adaptively find the relationship between two features and combine them by attention weight. Specifically, $\mathcal{H}_{t-1}$ is fed into two linear layers to produce K attention weights A_k and sampling offsets Δg_k. Later, based on the offsets Δg_k and coordinates g_k of reference points in $\mathcal{F}_t$, a bilinear interpolation is used to gather reference values G_k from $\mathcal{F}_t$. Finally, G_k is weighted by A_k to get enhanced feature $\hat{\mathcal{H}}_t$. The above process can be formulated as follows:

$$A_k = \text{Softmax}(\text{Linear}(\mathcal{H}_{t-1})),\ \Delta g_k = \text{Linear}(\mathcal{H}_{t-1}) \tag{3}$$

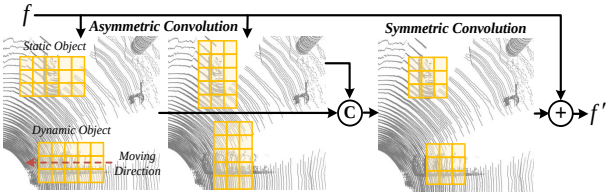
$$G_k = S(\mathcal{F}_t, g_k + \Delta g_k) \tag{4}$$

$$\hat{\mathcal{H}}_t = \sum_{l=1}^{L} W_l(\sum_{k=1}^{K} A_{lk} \cdot G_{lk}) \tag{5}$$

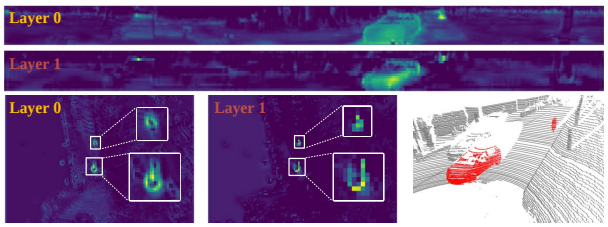
The L and K are the number of attention heads and reference points, respectively, while $S(\cdot)$ and W_l represent the bilinear sampling and learnable weight of multi-head attention. Next, the $\hat{\mathcal{H}}_t$ is processed by normalization layer and feed-forward network (FFN) to generate a updated $\mathcal{H}_t$ at the current time:

$$\tilde{\mathcal{H}}_t = \text{LN}(\hat{\mathcal{H}}_t + \mathcal{H}_{t-1}),\ \mathcal{H}_t = \text{LN}(\text{FFN}(\tilde{\mathcal{H}}_t) + \tilde{\mathcal{H}}_t) \tag{6}$$

Here, LN is the layer normalization. Then, $\mathcal{H}_t$ is used for two purposes: it replaces $\mathcal{H}_{t-1}$ to update the short-term memory, and it is fed into the decoder to predict segmentation results.



(a) The procedure of Asymmetric Convolution Block



(b) The visualization of BEV and range-view features at different multi-view encoder layers

Fig. 3. **Illustration of asymmetric convolution and multi-view features.**

D. Reduced-parameter Segmentation Decoder

To distinguish the static and dynamic points, the previous methods [7], [5], [8] usually leverage a UNet-like decoder to upsample multi-scale features progressively by convolutions. However, to reduce complexity and storage costs of network, we introduce a lightweight decoder, which first employs bilinear interpolation to convert the size of multi-scale features $\mathcal{G}_t^i$ $(i \in 1, 2)$ and $\mathcal{H}_t$ into a uniform height $H^b/2$ and width $W^b/2$. For each upsampled feature, we employ an auxiliary head to predict moving objects in BEV and exploit auxiliary loss as the constraint, which can guarantee that features from different scales are aligned and decoded well. Next, we adopt B2P to convert upsampled features into point features one by one and combine them to get $\hat{\mathcal{E}}_t \in \mathbb{R}^{V\times D}$. Finally, in addition to decoding the coarse motion states $\mathcal{C}_t \in \mathbb{R}^{V\times 3}$ of LiDAR points $\mathcal{P}_t$, the point-wise decoder also outputs the probability $\mathcal{O}_t \in \mathbb{R}^{V\times 2}$ that points belong to movable objects (e.g., cars, bicycles) and static backgrounds (e.g., roads), represented as:

$$\mathcal{C}_t = \text{head}_1(\text{Conv}(\hat{\mathcal{E}}_t) \odot \mathcal{E}_t), \mathcal{O}_t = \text{head}_2(\text{Conv}(\hat{\mathcal{E}}_t) \odot \mathcal{E}_t) \tag{7}$$

where head_1 and head_2 consist of several convolution blocks, and $\mathcal{E}_t$ is the feature from point-wise encoder. According to discrete classification labels $\mathcal{O}_t$, we can acquire the attributes of instance, like location and size, through clustering and use them to optimize $\mathcal{C}_t$ in the subsequent voting stage.

E. Long-term Voting Mechanism

Most existing approaches [4], [5], [19] focus on improving the quality of a single inference through modifying network structure. Nevertheless, in light of the inexplicability and data dependency of neural networks, this strategy may be limited. For example, for a parked car shown in Fig. 1(c), the model may predict it as stationary in one frame and moving in other frames. Meanwhile, due to lacking instance-level perception ability, the network may generate inconsistent results for the different parts of an object, particularly for cars (see Fig. 5).

To solve these problems, we construct a long-term memory bank of length M to store historical predictions and propose a voting module consisting of the voxel-based voting (VBV) and instance-based voting (IBV), which can function as post-processing to correct errors in the current predicted labels $\mathcal{C}_t$

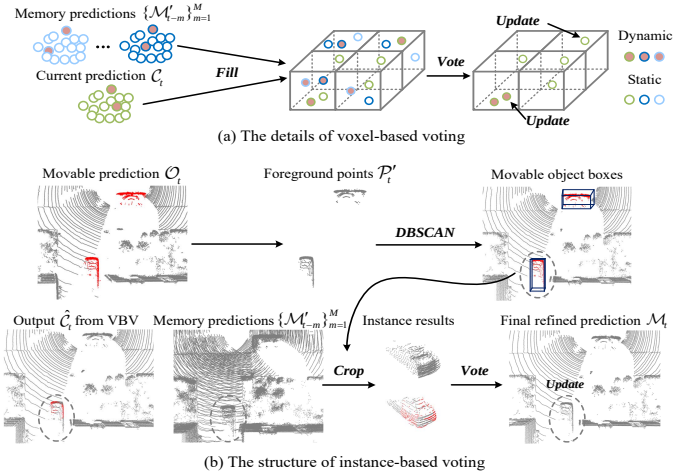


Fig. 4. **The details of our voting mechanism.** It uses voxel-based voting (VBV) and instance-based voting (IBV) to refine coarse predictions.

using historical results $\{\mathcal{M}_{t-m}\}_{m=1}^{M}$ and movable labels $\mathcal{O}_t$. Note that $\mathcal{M}_{t-m}$ $(m = 1, ..., M)$ with coordinates of $\mathcal{P}_{t-m}$ is projected into the coordinate system of $\mathcal{P}_t$ to yield $\mathcal{M}'_{t-m}$ through the pose transformations $\mathcal{T}_{t-m\to t}$ in advance.

1) Voxel-based voting: Motivated by TFNet [23], we first obtain historical predictions $\{\mathcal{M}'_{t-m}\}_{m=1}^{M}$ and current forecast $\mathcal{C}_t$ in the same coordinate system. Then, we divide points $\mathcal{P}_t$ into voxels with fixed size and fill $(\mathcal{C}_t, \{\mathcal{M}'_{t-m}\}_{m=1}^{M})$ into each voxel. Next, as shown in Fig. 4(a), the most frequently predicted label acts as motion state for all points in the same voxel and incorrect labels will be updated. We summarize the above procedure of VBV as: $\Omega(\mathcal{C}_t, \{\mathcal{M}'_{t-m}\}_{m=1}^{M}) \mapsto \hat{\mathcal{C}}_t$.

2) Instance-based voting: Although VBV can ensure the consistency of motion states within a local area and achieve performance improvement in Tab. IV, it is difficult to achieve instance-level unity, as shown by the output $\hat{\mathcal{C}}_t$ from VBV in Fig. 4(b). To solve this, we propose an instance-based voting based on clustering. When given the predicted probability $\mathcal{O}_t$ from decoder, we can pick out the foreground points $\mathcal{P}'_t$ from $\mathcal{P}_t$ and adopt DBSCAN [24] to split $\mathcal{P}'_t$ into S clusters. Then, according to the coordinates of points in each cluster, we can compute S minimum 3D bounding boxes to cover all objects. Thus, we can further crop out instance-level predictions from $\hat{\mathcal{C}}_t$ and memory predictions $\{\mathcal{M}'_{t-m}\}_{m=1}^{M}$. Finally, similar to voxel-based voting, we adopt the class label with the highest quantity as the motion state for all points in the instance and get the final prediction as: $\Phi(\hat{\mathcal{C}}_t, \{\mathcal{M}'_{t-m}\}_{m=1}^{M}, \mathcal{O}_t) \mapsto \mathcal{M}_t$.

Finally, when a new refined prediction $\mathcal{M}_t$ is output from voting mechanism, we append it to long-term memory while discarding the oldest result $\mathcal{M}_{t-M}$. As a result, compared to relying on the network's adaptive learning, voting mechanism can explicitly suppress incorrect predictions and improve the consistency of segmentation based on a statistical analysis of historical predictions at the voxel and instance levels.

F. Loss Functions

To ensure the network can be fully optimized, we separate the training process into two steps. In the first stage, we only train our network without predicting movable objects in the decoder. Meanwhile, following the previous works [7], [5], we introduce the weighted cross-entropy (L_{wce}) and Lovász-Softmax (L_{ls}) [25] losses to supervise network:

$$L = \lambda_1 L_{wce} + \lambda_2 L_{ls}, \;\; L_{s1} = L(y, \hat{y}) + \lambda_3 \sum_{i=1}^{3} L(y_i^b, \hat{y}_i^b) \quad (8)$$

where λ_1, λ_2, λ_3 mean the weights for losses while y and $\hat{y}$ are the ground truth and predicted results of points. $L(y_i^b, \hat{y}_i^b)$ denotes the auxiliary losses for BEV predictions. Moreover, in the second stage, we freeze pre-trained parameters that are optimized in the 1^{st} stage and only train the rest of network to predict movable objects using the following loss function:

$$L_{s2} = \lambda_1 L_{wce}(x, \hat{x}) + \lambda_2 L_{ls}(x, \hat{x}) \quad (9)$$

where x and $\hat{x}$ represent ground-truth labels and predictions for movable objects.

IV. Experiments

A. Experimental Settings

Datasets. On SemanticKITTI-MOS [4] dataset and Sipailou-Campus [7] dataset, we compare segmentation performance with previous methods and conduct extensive ablation studies. The SemanticKITTI-MOS dataset is collected by a Velodyne HDL-64E LiDAR and contains a total of 22 sequences with labeled point clouds that are remapped from 28 semantic classes into 3 types of motion states. Following the previous algorithms [7], [5], [8], we divide the sequences 00-07, 09-10 for training, sequence 08 for validation and sequences 11-21 for testing. For the Sipailou Campus dataset that is developed on solid-state LiDAR, we follow the implementation of [7] to split dataset into 5 training sequences, 1 validation sequence and 2 test sequences from 26,279 frames.

Evaluation Metric. Consistent with present approaches [8], [5], we employ the Jaccard Index or Intersection-over-Union (IoU) metric [26] over dynamic objects to measure the MOS performance, which can be denoted as:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (10)$$

where TP, FP, and FN mean the number of true positive, false positive, and false negative predictions for dynamic category.

B. Implementation Details

In data processing, we leverage widely used data augmentation, such as random rotation, flipping and slight translation to enrich the training data, which plays an important role in improving model generalization. Meanwhile, as mentioned in Sec. III-F, we optimize the network using two-stage training strategy. For the 1^{st} stage, we train the model for 48 epochs on NVIDIA RTX 4090 GPUs using an SGD optimizer with an initial learning rate of 0.02, which is decayed by 0.1 every 10 epochs. For the 2^{nd} stage, we solely optimize the network for 10 epochs with a learning rate of 0.02. Furthermore, each LiDAR scan is limited to [-50m, 50m] for the X and Y axes and [-4m, 2m] for the Z axis. The number of points in each scan is randomly downsampled or padded to $V = 1.3 \times 10^5$. The default values for the number of attention heads L and reference points K are both set to 4.

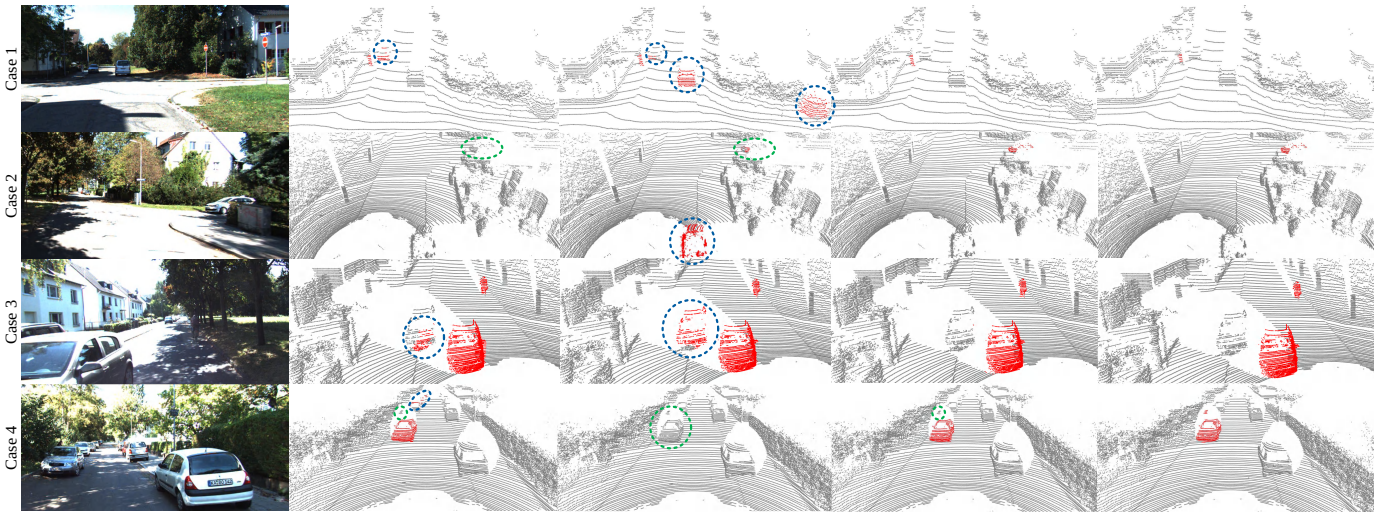


Fig. 5. **The visualization of MOS results on the SemanticKITTI validation set.** Incorrect predictions are highlighted, with false negatives marked by green circles and false positives by blue circles. Best viewed in color and zoom.

TABLE I
PERFORMANCE COMPARISON ON SEMANTICKITTI VALIDATION AND TEST SETS. * DENOTES METHODS THAT EXPLOIT SEMANTIC LABELS. † INDICATES METHODS TRAINED ON BOTH SEMANTICKITTI AND KITTI-ROAD DATASETS.

Methods	Source	IoU (Validation)	IoU (Test)
KPConv	*ICCV 19*	-	60.9
SpSequenceNet	*CVPR 20*	-	43.2
LiMoSeg	*arXiv 21*	52.6	-
LMNet	*RA-L 21*	66.4	58.3
Cylinder3D	*CVPR 21*	66.3	61.2
AutoMOS	*RA-L 22*	-	54.3
MotionSeg3D, v1	*IROS 22*	68.1	62.5
MotionSeg3D, v2	*IROS 22*	71.4	64.9
4DMOS	*RA-L 22*	77.2	65.2
MotionBEV, w/o delay	*RA-L 23*	68.1	63.9
MotionBEV, w/ delay	*RA-L 23*	76.5	69.7
StreamMOS-V	-	**78.3**	**73.1**
LMNet*	*RA-L 21*	67.1	62.5
RVMOS*	*RA-L 22*	71.2	74.7
InsMOS*	*IROS 23*	73.2	70.6
InsMOS*†	*IROS 23*	-	75.6
MF-MOS*	*ICRA 24*	76.1	76.7
StreamMOS-VI*	-	**81.6**	**77.8**

TABLE II
PERFORMANCE COMPARISON ON SIPAILOU CAMPUS DATASET.

Methods	Source	IoU (Validation)	IoU (Test)
LMNet	*RA-L 21*	54.3	56.2
MotionSeg3D, v2	*IROS 22*	65.6	66.8
4DMOS	*RA-L 22*	87.3	88.9
MotionBEV	*RA-L 23*	89.2	90.8
StreamMOS-V	-	**90.9**	**92.5**

TABLE III
COMPARISON OF RUNNING TIMES (MS) WITH PREVIOUS METHODS.

4DMOS	MF-MOS*	MotionSeg3D, v1	MotionSeg3D, v2
86	96	42	117
InsMOS*	RVMOS*	**StreamMOS-V**	**StreamMOS-VI***
120	29	62	96

C. Quantitative Results

Comparison with Previous Methods. We first evaluate our StreamMOS on SemanticKITTI-MOS benchmark. To ensure fairness, our method is presented in two versions in Tab. I to make settings as consistent as possible with previous works. Specifically, (a) *StreamMOS-V* indicates the network that is trained in the 1^{st} stage and uses voxel-based voting as post-processing. (b) *StreamMOS-VI** means performing extra 2^{nd} stage training and using instance-based voting that relies on movable object predictions. Then, our methods are compared with existing algorithms, which can be classified as whether semantic annotations are utilized. Specially, the two versions of MotionSeg3D [5] refer to using kNN or point refinement as post-processing. Moreover, "w/ delay" signifies exploiting point cloud frames within the time window of $[t, t+N]$ to estimate dynamic objects in the t frame. Note that following MotionBEV [7], our results shown in Tab. I are derived from training original SemanticKITTI without any additional data.

As illustrated in Tab. I, our streaming method outperforms previous works in most cases. Specifically, our StreamMOS-V exceeds 4DMOS [9] by 1.1% and 7.9% in validation and test. We think that compared with using binary Bayes filter to merge historical results in 4DMOS, our method additionally considers historical feature from the last inference, which can serve as strong spatial priors to improve prediction quality. At the same time, our StreamMOS-VI* surpasses InsMOS* [8] and MF-MOS* [27] on the validation set significantly (↑8.4% and ↑5.5%) by instance-based voting. Finally, due to the lack of semantic annotation in Sipailou-Campus dataset, we solely list [4], [7], [5], [9] in Tab. II and confirm the effectiveness of StreamMOS-V, even when using a solid-state LiDAR with a narrow field of view and non-repetitive scanning patterns.

Inference Speed. Although our method uses attention mechanism to construct feature association between inferences and merge multiple historical predictions in voting mechanism, it still keeps competitive running time compared with previous approaches in Tab. III. We believe this is the contribution of projection-based backbone, lightweight deformable attention and parameter-free upsampling in decoder, which enables our method strike a balance between speed and performance.

D. Qualitative Analysis

Advantageous Cases. In Fig. 5, we exhibit the segmentation results in various scenarios to compare the previous methods

TABLE IV
THE EFFECT OF DIFFERENT MODULES IN SEMANTICKITTI VALIDATION.

	TF	MVE	VBV	IBV	IoU [%]	Δ
A1					67.1	-
A2	✓				73.2	+6.1
A3	✓	✓			77.1	+10.0
A4	✓	✓	✓		78.3	+11.2
A5	✓	✓		✓	81.3	+14.2
A6	✓	✓	✓	✓	**81.6**	+14.5

TABLE V
ABLATION EXPERIMENT ON MULTI-VIEW ENCODER OF STREAMMOS-V.

	RV	BEV	ACB	Parallel	Series	IoU [%]
B1	✓					70.3
B2		✓				74.2
B3	✓	✓			✓	77.5
B4	✓	✓		✓		74.8
B5	✓	✓	✓		✓	**78.3**

TABLE VI
ABLATION EXPERIMENT ON TEMPORAL FUSION OF STREAMMOS-V.

	Strategy	IoU [%]	Δ
C1	w/o Temporal Fusion	72.1	-6.2
C2	Cross-attention	73.0	-5.3
C3	Concatenation	74.8	-3.5
C4	Addition	75.6	-2.7
C5	Deform-attention	**78.3**	-

intuitively. Although MotionSeg3D adopts a point refinement module to alleviate boundary-blurring problem, it still makes mistakes when dealing with distant objects, as shown in the 4^{th} row. Besides, MotionSeg3D tends to produce incomplete segmentation in the 3^{rd} row due to lacking the instance-level sensing. Despite adding instance detection like InsMOS* can improve segmentation integrity, it aggravates negative impact when the prediction is incorrect, as illustrated in the 1^{st}, 2^{nd} and 3^{rd} rows. Unlike these algorithms, our StreamMOS-VI* combines multi-view observations to improve the perception of objects at different distances. Then, we build relationships among several inferences by integrating memory feature and predictions to enhance the segmentation integrity and reduce incorrect results. Thus, we get superior performance in Fig. 5.

Failure Cases. Fig. 6 displays that the inaccurate ego poses $\{\mathcal{T}_{t-n\to t}\}_{n=1}^{N}$ misalign multi-frame point clouds, causing the network to incorrectly infer that the object has moved. While our method mitigates this issue compared to InsMOS*, errors persist as our voting mechanism still relies on precise poses. Thus, we think that developing a MOS model that eliminates the need for pose or implicitly learns ego-motion could be a promising research direction in the future.

E. Ablation Study

This part conducts ablation studies on the SemanticKITTI validation set to prove the effectiveness of our method.

Model Components. As shown in Tab. IV, our StreamMOS mainly includes some crucial modules: temporal fusion (TF), multi-view encoder (MVE), voxel-based voting (VBV), and instance-based voting (IBV). To understand their importance in overall performance, we first remove all the above modules from our StreamMOS and regard the rest as a baseline in A1. After building feature correlations between inferences by TF, the IoU increases by 6.1% in A2. Moreover, benefiting from capturing multi-view motion cues from BEV and RV, MVE brings further improvement. Then, due to introducing object-level perception, instance-based voting in A5 shows a greater performance than voxel-based in A4, which only focuses on limited areas in the 3D cube. Finally, we can achieve optimal performance by combining them into a refinement procedure from voxel to instance, proving that effectively utilizing long-term predictions is the key element to the LiDAR MOS task.

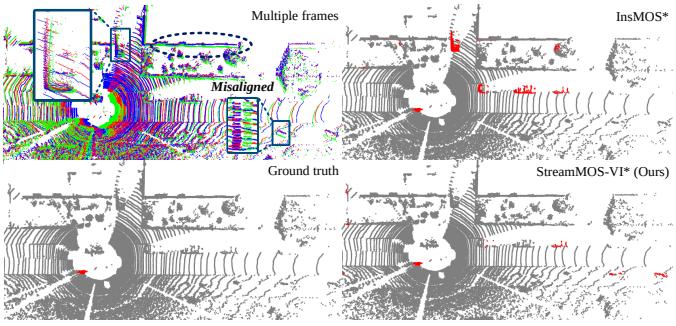


Fig. 6. **A failure case caused by inaccurate ego pose in SemanticKITTI.**

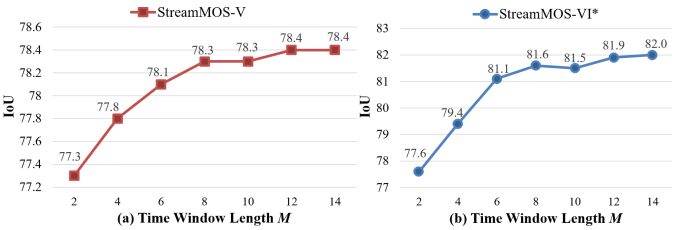


Fig. 7. **Ablation study on the time window length of voting mechanism.**

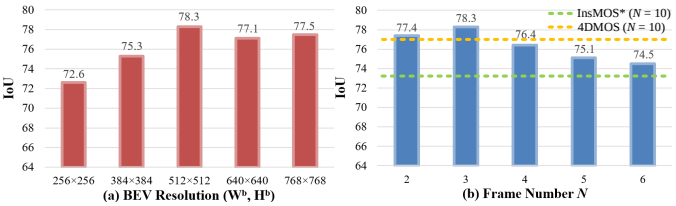


Fig. 8. **The effect of frame number and BEV size on the SreamMOS-V.**

Multi-view Encoder. We compare several multi-view encoding strategies in Tab. V. From the B1 and B2, we can observe that when encoding object motion only on a single view, the BEV representation achieves better results compared to RV due to global perspective and motion consistency. Then, we divide encoder into BEV and RV branches and extract multi-view features in series (B3), leading IoU to further increase and exceed parallel mode (B4) by 2.7%. We think that series manner may be more suitable for deriving consistent moving features from different views owing to progressive encoding. Furthermore, using asymmetric convolution block (ACB) can result in 0.8% improvement in B5, proving the advantage of decoupling horizontal and vertical encoding.

Temporal Fusion. The strategy of propagating the historical feature into current inference will affect segmentation quality as demonstrated in Tab. VI. First, we can observe that lacking temporal fusion to provide prior information leads to unideal results (↓6.2%). Then, compared with adopting concatenation and addition directly to merge features in different coordinate systems, deformable attention could align features adaptively by learnable offsets and gain the advantage of 3.5% and 2.7% IoU. Moreover, it is worth noting that cross-attention gets the

worst result since redundant global attention may cause a bad effect. In contrast, deformable attention concentrates on local feature to avoid model overfitting and save computation load.

Time Window Length. The time window length determines how long ago predictions can be used by voting mechanism. Thus, we conduct experiments on the time window length to choose the optimal setting for our algorithm. As displayed in Fig. 7, the performance will increase rapidly until the length M of time window reaches 8. Despite continuing to raise the length could result in a slight improvement, it requires more time consumption. Thus, we opt for $M = 8$ as our default.

Other Hyper-parameter Settings. In Fig. 8, we explore the impact of frame number and BEV resolution on performance. We can observe that the optimal BEV size (W^b, H^b) is 512×512. Meanwhile, too small BEV resolution would cause the network to be unable to capture the motion of small objects, while excessively large resolution leads to sensitivity to slight disturbances. Besides, a larger BEV image will contain more numerous empty grids, which may dilute useful information.

Furthermore, as shown in Fig. 8(b), compared to previous approaches [8], [9] that require a lot of frames to extract the spatial-temporal features, our method only relies on 3 frames to achieve the best result. We think this is due to the effective reuse of historical feature and predictions in temporal fusion and voting, which bring rich prior knowledge to the network. Meanwhile, feeding too many frames into the network would cause information redundancy and result in degradation.

V. Conclusion

In this paper, we analyze the limitations of existing MOS methods and propose a novel streaming structure, which uses memory bank as a bridge to transfer prior information among inferences while capturing the appearance and motion feature of objects from multiple views. To correct false predictions, we propose a voting mechanism to integrate historical results at the voxel and instance levels. Experimental results indicate that our method performs competitively in diverse aspects.

References

[1] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and systems*, vol. 2, pp. 1–9, Berkeley, CA, 2014.

[2] B. Guo, N. Guo, and Z. Cen, "Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5850–5857, 2022.

[3] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," *Neurocomputing*, vol. 497, pp. 64–75, 2022.

[4] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[5] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen, "Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11456–11463, IEEE, 2022.

[6] S. Mohapatra, M. Hodaei, S. Yogamani, S. Milz, H. Gotzig, M. Simon, H. Rashed, and P. Maeder, "Limoseg: Real-time bird's eye view based lidar motion segmentation," *arXiv preprint arXiv:2111.04875*, 2021.

[7] B. Zhou, J. Xie, Y. Pan, J. Wu, and C. Lu, "Motionbev: Attention-aware online lidar moving object segmentation with bird's eye view based appearance and motion features," *IEEE Robotics and Automation Letters*, 2023.

[8] N. Wang, C. Shi, R. Guo, H. Lu, Z. Zheng, and X. Chen, "Insmos: Instance-aware moving object segmentation in lidar data," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7598–7605, IEEE, 2023.

[9] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss, "Receding moving object segmentation in 3d lidar data using sparse 4d convolutions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7503–7510, 2022.

[10] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE robotics and automation letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[11] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla, "Robust method for removing dynamic objects from point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10765–10771, IEEE, 2020.

[12] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3712–3719, 2014.

[13] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10758–10765, IEEE, 2020.

[14] R. Ambruş, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1854–1861, 2014.

[15] H. Lim, S. Hwang, and H. Myung, "Erasor: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3d point cloud map building," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2272–2279, 2021.

[16] J. Zhang and Y. Zhang, "Erasor++: Height coding plus egocentric ratio based dynamic object removal for static point cloud mapping," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4067–4073, 2024.

[17] H. Lim, L. Nunes, B. Mersch, X. Chen, J. Behley, H. Myung, and C. Stachniss, "Erasor2: Instance-aware robust 3d mapping of the static world in dynamic scenes," in *Robotics: Science and Systems (RSS 2023)*, IEEE, 2023.

[18] T. Kreutz, M. Mühlhäuser, and A. S. Guinea, "Unsupervised 4d lidar moving object segmentation in stationary settings with multivariate occupancy time series," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1644–1653, 2023.

[19] J. Kim, J. Woo, and S. Im, "Rvmos: Range-view moving object segmentation leveraged by semantic and motion features," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8044–8051, 2022.

[20] X. Li, G. Zhang, H. Pan, and Z. Wang, "Cpgnet: Cascade point-grid fusion network for real-time lidar semantic segmentation," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 11117–11123, IEEE, 2022.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," 2021.

[23] R. Li, S. Li, X. Chen, T. Ma, J. Gall, and J. Liang, "Tfnet: Exploiting temporal cues for fast and accurate lidar semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4547–4556, 2024.

[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, pp. 226–231, 1996.

[25] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4413–4421, 2018.

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[27] J. Cheng, K. Zeng, Z. Huang, X. Tang, J. Wu, C. Zhang, X. Chen, and R. Fan, "Mf-mos: A motion-focused model for moving object segmentation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12499–12505, 2024.